Client/Server Application Development

Client/Server Application Development is simply the process of planning how a client/server system and its applications will be built. It begins with modeling on paper all the entities that may impact a system and continues through prototyping, revising and final implementation.

Application development has been performed for decades but today's client/server and web paradigms bring unique challenges to the design process. Perhaps the greatest challenge is the speed with which technology changes. New versions of hardware and software are introduced weekly as opposed to monthly or yearly. It seems that once development is complete, the system is technically obsolete.

Ultimately, the purpose of information technology and the development process is to automate the business model. The following quote is from a March, 1997 paper published by IDC, “Strategic Fit of Enterprise Servers in Large Organizations.”

“One factor mentioned by many CIOs was the importance of not forcing the business organization to fit technology. As one said, ‘If you split the applications and database by geography this year, you can be sure that the organization will want to organize by product next year. Explaining why IT is a constraint to such a reorganization is a career limiting opportunity.’”

So what does a development team do in order to keep up with the pace of change? Teams use open systems, standard hardware and software, minimize proprietary technology, and they select vendors who are capable of long term support and upgrades.

In this paper we will take a brief look at the process of application development, explore the issues with which designers wrestle and discuss several techniques used by IT professionals to try to manage today’s challenges.

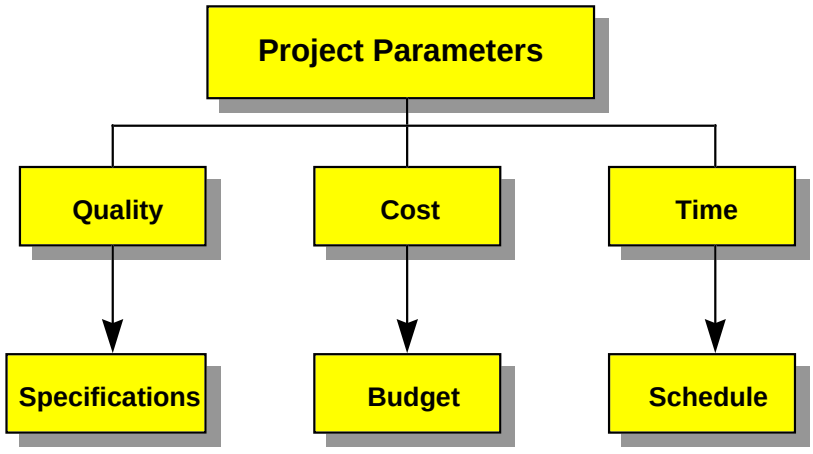
A Few Words About Project Management

Client/Server application design is at its core a project management function. Project management differs from other management principles in two significant ways: first it focuses on an undertaking with a finite life span, and second, frequently uses resources on a part-time basis, vs. permanent operations that use resources on a full-time basis.

All projects move through a predictable lifecycle of four phases:

- Conceiving and defining the project
- Planning the project
- Implementing the plan
- Completing and evaluating the project

During a project's life, system planners must focus on three basic parameters: quality, cost and time. These three parameters form a fixed equation where a change in one parameter MUST be offset by a change in one or both of the remaining parameters. For example if time is reduced then quality must go down or cost must increase. A successful project is one that is completed at the specified level of quality, on or before the deadline, and within budget.



Of course in reality, specifications change over the life of a project, which will impact both schedules and budgets. It is the project manager's responsibility to ensure that the client is made aware of and agrees to any revisions.

Life Cycle of an Information Technology System

Over the years systems designers have come to realize that client/server systems have a life of their own. Most systems life cycles will go through something like these eight steps:

1. Feasibility Study - A conceptual phase. Decisions are made based on questions such as:

   - What value will the new system offer?
   - What will it cost?
   - What resources are required?
   - Can it be done in a set timeframe?

   The feasibility study seeks to put hard data into project management guidelines of cost, schedule and quality.

2. Analysis - This is the phase in which the computer system's capability requirements are defined and prioritized. Tradeoffs, risks and technology shortfalls are identified. A summary of ideas that will not be implemented should be identified as well. It does not consider *how* the new system will perform it duties. This is the most critical phase and will ultimately make or break the success of the project and management team. Do not pass this step until all key players are aware of the capabilities that will and will NOT be implemented.

3. Design - Design is the process of creating a plan for implementing the specifications of a new system. This is the "how " phase. A complete project analysis is done at this phase to define the project equation of quality, cost, and time. Maintenance, upgrades and obsolescence details are included here.

4. Implementation - This is the phase in which the software programs that make up the new system are coded and tested, and the hardware is purchased and installed.

5. Acceptance Testing - Just as it sounds. All pieces of the system are tested to ensure that they work according to the design specifications.

6. Production - Provided all previous phases are completed successfully, production begins and users start to utilize the system.

7. Maintenance - As soon as production begins, maintenance quickly follows. Maintenance corrects bugs within the system that were not identified in earlier phases. Maintenance may also reconfigure the system to perform new functions. One challenge is to ensure that any changes made do not create more problems than they solve.

8. Obsolescence - Eventually all systems become obsolete. Hopefully with careful planning the computer system will have performed well enough and long enough to ensure an acceptable return on investment.

Not all projects will go through all eight steps. In the real world some steps are skipped, merged or are implemented simultaneously.

Layered Architectures

One approach to designing client/server systems is to focus on clearly defined layers of the application architecture. This approach is similar to the way architects design and plan the construction of a building. Each building is designed using blueprints that show the layout of various "layers" of the building. There is a blueprint for electrical circuitry, the plumbing, and structural components.

Like layers in a building, client/server system layers can be designed independently by specialists provided that the connections or interfaces between the layers are carefully planned. Each layer must remain independent of the other layers with clearly defined domains and protocols for interacting with one another.

The layers should not be tied directly to the specific application. For example, an architecture might work equally well for both a payroll system and an on-line ordering system. Let us take a quick look at two types of software design methodologies: the two-layer and three-layer architectures (also known as two-tier and three-tier).

Two-Layer Architecture

The two-layer architecture consists of the Application Layer and the Database Layer

Application Layer - This layer handles presentation, application and transaction logic. Presentation logic is the Graphical User Interface (GUI) with which the user interacts with the application. The application logic handles the business rules and policies (such as what to do if a customer exceeds her credit limit). The transaction logic is the code that groups database updates into transactions and ensures that all updates within a transaction are made consistently.

Database Layer - This layer consists of the underlying database engine that supports the application. It is responsible for maintaining the integrity of the database. Some or all of the transaction logic may be implemented in this layer.

In the two-layer model, the application layer is usually implemented with a visual-programming tool. The database layer is built with a database management system such as Oracle 7 or Microsoft SQL Server.

The distinctions between the application layer and the database layer are not always well defined. For performance reasons the transaction logic may be shifted to the database layer in the form of stored procedures or triggers. Thus the database layer is often handling some of the application logic. The solution is to create a third layer that will handle the transaction and business rule logic.

Three-Layer Architecture

In the Three-Layer Architecture an additional layer is added to handle transaction logic, and the top layer has sole responsibility for the user interface.

User Interface Layer - This is where the design of the GUI present logic is performed.

Business Rule Layer - Rules and Policy logic unique to the application or organization are designed in this layer.

Database Layer - This layer is responsible for maintaining the integrity of the database. It is the underlying database model that supports the application.

The User Interface

Much of the success of a client/server application depends upon the design of the user interface. The interface is *the application* to the end user. The individuals using the system do not care if you used a two or three-layer architecture or how the database was configured.

User interface design can quickly become one of the most complicated aspects of a client/server application. The interface may include menu commands, toolbars, buttons, keyboard shortcuts, drag and drop features, slider bars, and other controls. The designer must decide what forms and fields are required, what controls to present to the user and in what sequence to present the features.

Joint Application Design

Joint Application Design is an approach in which all parties which may be involved at some point in the development of the application are brought together to work through the various issues. This team should include designers, a high level manager, end users that have a solid understanding of the business process that the application addresses, and technical specialists.

Rapid Application Development

Rapid Application Development attempts to shorten the overall development cycle through the use of prototyping and overlapping steps in the development process. Analysis, design, implementation and testing are all done simultaneously in incremental steps.

Rapid Application Development is implemented using the Joint Application Design team creating a working prototype of the system. The prototype is then reviewed, refined, and reviewed again. Typically the tools used to create the prototype are the same tools which will be used to create the final product.

Prototyping creates the shell of an application, giving the illusion of a complete or nearly complete program. The prototype may not take into account all of the possible scenarios in which the final product will need to operate.

Conclusion

Client/Server application development is a huge subject that can easily fill volumes. We have attempted to give you a broad outline of some of the major approaches, techniques and issues which information technology professionals must consider when designing a client/server application.

Key Points to Remember

- Three parameters of any system analysis and design project are quality, cost and time.
- A Two-Layered Architecture consists of the Application Layer and the Database Layer.
- A Three-Layered Architecture consists of the User Interface Layer, the Business Rule Layer, and the Database Layer.
- End users should be included in the Analysis and Design process, particularly when designing the user interface.
- Rapid Application Development shortens the development cycle through the use of prototyping and by overlapping steps of the development process.